09｜RTP & RTCP：如何正确地将视频装进RTP中？

2021-12-10 李江

《攻克视频技术》 课程介绍 >

讲述：李江
时长 16:22 大小 14.99M

你好，我是李江。

在前面的课程中，我们详细地讲述了视频编码的原理以及预测编码和变换编码的知识。通过这些我们了解了视频编码的基本原理和步骤。同时，我们还用了一节课的时间深入探讨了 H264 的码流结构，相信你已经清楚了 H264 码流是什么样的，以及如何从码流中分离出一帧帧图像数据，并学会了如何判断这些帧的类型。

那么从这节课开始呢，我们就要进入视频传输和网络对抗部分了。我们会在视频编码码流的基础上，讲讲如何将码流打包成一个个数据包发送到网络上，并进一步讨论如何避免发送的过程中引起网络拥塞，从而保证视频的流畅性。同时，我们会进一步在后面的课程中讲解如何在网络不断变化的时候做好视频码控算法，如何防止视频出现花屏，以及如何尽量减少视频卡顿等非常有难度的实际工程问题。

这些问题是视频开发过程中经常会遇到且迫切需要解决的重要问题。而解决这些问题的基础就是需要熟悉 RTP 和 RTCP 协议，也就是我们这节课的重点。

接下来我们会分别从 RTP 协议、RTCP 协议和 H264 的 RTP 打包方法这三个方面来展开这节课。首先让我们一起来认识一下 RTP 协议。

RTP 协议

RTP（Real-time Transport Protocol）协议，全称是实时传输协议。它主要用于音视频数据的传输。那它的作用是什么呢？

一般我们在实时通信的时候，需要传输音频和视频数据。我们通常是这样做的，先将原始数据经过编码压缩之后，再将编码码流传输到接收端。在传输的时候我们通常不会直接将编码码流进行传输，而是**先将码流打包成一个个 RTP 包再进行发送**。

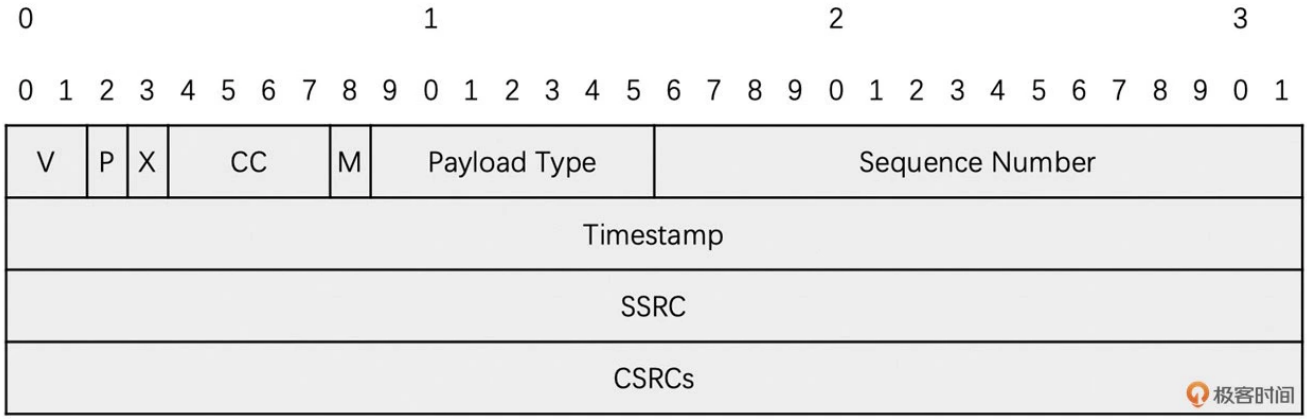
那为什么需要打包成 RTP 包呢？这是**因为我们的接收端要能够正确地使用这些音视频编码数据，不仅仅需要原始的编码码流，还需要一些额外的信息**。比如说：

> **当前视频码流是哪种视频编码标准**，是 H264、H265、VP8、VP9 还是 AV1 呢？我们知道每种不同的编码标准，其码流解析的方式肯定也不一样。这个就需要通过 RTP 协议告知接收端。
>
> 当我们知道编码标准了，我们就可以正确地解析码流，并解码出图像了。但是我们又会遇到一个新的问题，那就是**按照什么速度播放视频呢？**这个也需要 RTP 协议告知接收端。
> 这就是 RTP 协议的一个重要的作用，即告知接收端一些必要的信息。当然 RTP 协议的作用不止这些，它其实在网络带宽预测和拥塞控制的时候也发挥出了至关重要的作用。我们在之后的课程中会继续讨论，这里就先不讲了，你大体有个印象就可以。

我们知道 RTP 包需要附带很多额外的信息，那这些信息在 RTP 包中是怎么存在的呢？其实 RTP 包包括两个部分：第一个部分是 RTP 头；另外一个部分是 RTP 有效载荷。其中 RTP 头主要是用来携带前面说的那些额外信息的，等会儿我会详细介绍一下 RTP 头部每个字段的意义。

这里我先稍微跟你解释一下另外一个部分，也就是 RTP 有效载荷。RTP 有效载荷，其实就是 RTP 包里面的实际数据。如果是 H264 编码打包成 RTP 包，那有效载荷就是经过 H264 编码的码流；如果是 VP8 编码呢，那就是 VP8 码流。

接下来，我们重点来看看 RTP 包的头部。具体如下图所示：



是不是有点懵，别急，下面我给了一张表格，你可以对照着表格看看 RTP 包头的每一个字段占用的位数和具体的含义。其中绿色部分是很重要的知识点，需要你重点掌握。

字段	占用位数	含义
V	2	RTP版本号。当前版本号必须是2。
P	1	填充标志位。当为1时，表示在RTP包的最后面有若干个填充字节。注意填充字节只用于字节对齐填充，不是视频数据。
X	1	扩展标志位。当为1时，表示在RTP默认头部后面会有一个用于自定义的扩展头部
CC	4	CSRC个数。表示CSRC标识符个数。
M	1	Marker位。对于不同的有效载荷Marker位含义不同。对于视频，表示当前RTP包是一帧的最后一个包。
Payload Type	7	有效载荷类型。用于说明RTP报文中有效载荷的类型，比如说可以是H264、VP8等，用于接收端进行解析RTP的有效载荷。
Sequence Number	16	序列号。用于表示当前RTP包的序号，每发送一个RTP包，序列号加1。它是RTP包的标识。接收端可以使用它来告诉发送端RTP包丢失了，要求发送端重传。序列号的初始值是随机的。
Timestamp	32	时间戳。时间戳反映了该RTP包所属视频帧的采样时间。接收端使用时间戳来控制播放速度和音视频同步。使用90000Hz的时间基（就是说时间戳的单位是1/90000秒）
SSRC	32	同步信源。比如说当前RTP用于传输H264视频数据和OPUS音频数据。H264的RTP包使用同一个SSRC，OPUS的RTP包使用同一个SSRC。接收端根据SSRC标识符来区分当前RTP包是H264 RTP包还是OPUS RTP包
CSRCs	每一个占用32位，数量由CC指定	提供信源，可以有0~15个CSRC。每个CSRC标识了包含在RTP包有效载荷中的所有提供信源。主要用在混合器中。比较少遇到。不展开讲。

上面讲的就是 RTP 头部的主要组成部分。在这里需要单独提一下 RTP 头部的另外一个比较重要的部分，就是 RTP 扩展头。从上表我们可以看到，RTP 包头有一个扩展头标志位 X，当扩展头标志位 X 为 1 的时候，说明有 RTP 扩展头。RTP 扩展头由于平时大家很少用看似不怎么重要，但是在 RTC 场景中，尤其是 WebRTC 中经常会用到。另外，RTP 扩展头我们在带宽预测的时候也会用到。所以建议你也了解一下。

扩展头主要是用来给用户自定义扩展使用的。因为协议是标准的，但是用户使用场景却是多种多样的，所以 RTP 需要考虑的比较全面，留了一个扩展头可以让用户根据使用场景和需求，自己定义扩展头，用来传输需要在 RTP 包中传输的信息。扩展头的格式可以参考这篇文章，这里就不做过多的展开了。

好了，以上就是 RTP 协议的主要知识点。有了 RTP 协议，我们就能够将码流打包成 RTP 包发给接收端了。如果你只负责传输 RTP 包，而不需要管传输过程中有没有丢包，以及传

输 RTP 包的时候有没有引起网络拥塞的话，那你只需要使用 RTP 协议就可以了。比如说，你选择使用 TCP 协议传输 RTP 包的话就可以不用管这些事情，因为 TCP 协议具有丢包重传、拥塞控制等功能。

但是通常情况下，我们在传输音视频数据的时候不会使用 TCP 协议作为传输层协议。这是因为 TCP 协议更适合传输文本和文件等数据，而不适合传输实时音频流和视频流数据，所以我们通常会**使用 UDP 协议作为音视频数据的传输层协议**。但 UDP 协议不具有丢包重传和拥塞控制的功能，需要我们自己实现。那怎么办呢？

其实，真要做好丢包重传和拥塞控制是非常难的，一节课也讲述不清楚，所以，我们会在接下来的好几节课里详细解释。接下来我们可以先关注下丢包重传和拥塞控制的基础之一，也就是 RTP 协议的“好兄弟”，RTCP 协议。

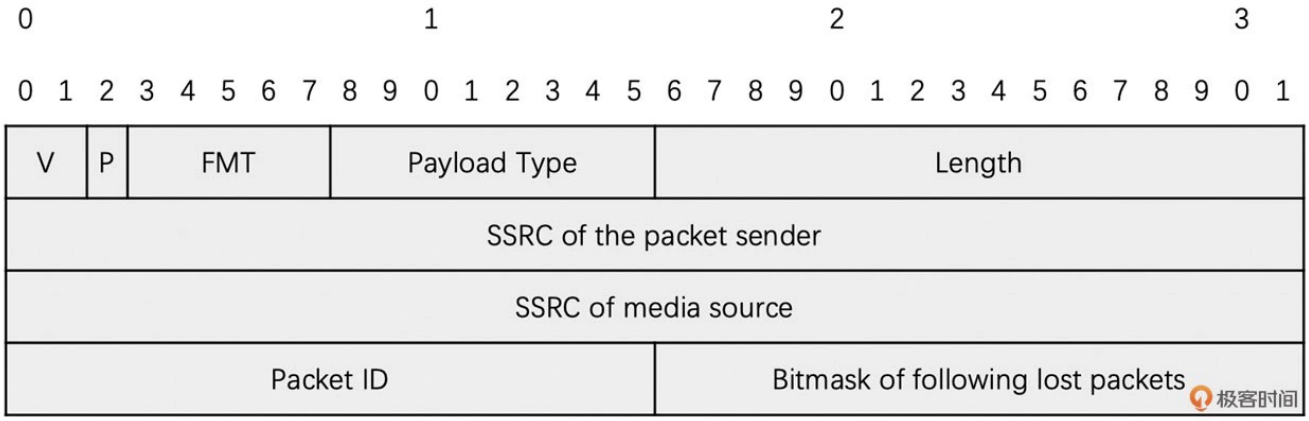
RTCP 协议

RTCP（Real-time Transport Control Protocol）协议，全称是实时传输控制协议。它是辅助 RTP 协议使用的。RTCP 报文有很多种，分别负责不同的功能。常用的报文有发送端报告（SR）、接收端报告（RR）、RTP 反馈报告（RTPFB）等。而每一种报告的有效载荷都是不同的。我们就是通过这些报告在接收端和发送端传递当前统计的 RTP 包的传输情况的。我们使用这些统计信息来做丢包重传，以及预测带宽。

不过，我需要再次强调一下，**RTCP 协议只是用来传递 RTP 包的传输统计信息，本身不具有丢包重传和带宽预测的功能，**而这些功能需要我们自己来实现。

我们上面讲到了 RTCP 协议有很多种报告，而每种报告其实定义的具体内容都是不一样的。我们这里以 RTPFB 报告中的 NACK 报告（丢包提示报告）作为一个例子来看看 RTCP 协议大概是什么样子的。（RTPFB 报告包含了多种子报告，NACK 报告只是其中的一种，因为我们后面还会用到这个报告，所以这里我们就先以这个报告为例子。）

下图就是 NACK 报告的协议格式。



其中，每一个字段在下表中都有详细的解释。

字段	占用位数	含义
V	2	RTP版本号。当前版本号必须是2。
P	1	填充标志位。当为1时，表示在RTP包的最后面是不是有若干个填充字节。注意填充字节只用于字节对齐填充，不是视频数据。
FMT	5	表示具体指的是哪种RTPFB消息，参考RFC4585。NACK报告的FMT为1。
Payload Type	8	RTCP报告的类型，参考RFC4585。RTPFB的值是205。
Length	16	当前RTCP报告的大小减去1，以4字节为单位。
Packet ID	16	当前RTCP报告中第一个丢失的RTP包的序列号
Bitmask of following lost packets	16	表示紧接着Packet ID的第i个包是不是丢失了，丢失了则第i位为1。

极客时间

从上面的 NACK 报告我们可以看到，RTCP 协议跟 RTP 不同，它们传递的东西是不一样的。

我们知道 RTP 是用来传输实际的视频数据的。它就像一个快递盒，先装好视频，然后填好运送的视频基本信息和收件人信息，最后将视频运送到收件人手上。

而 RTCP 协议则像是一个用来统计快递运送情况的记录表。其中的 NACK 报告就是快递丢件情况的记录表。它记录着哪些快递丢了。发件人收到了 NACK 之后，可以重新寄一个同样的快递给收件人，防止收件人没有收到快递。在这里也就是将丢失的视频 RTP 包重传一遍。

虽然我们只讲了一种 RTCP 报告，但是其它的报告也是类似的。大多数报告都是用来记录传输信息的。因为数量很多，我们这里就不一一展开了。如果你有兴趣的话，可以查看这个RFC 文档。

好了，通过学习 RTP 和 RTCP 的基础知识，我们了解了 RTP 包的协议格式和主要负责的功能，也知道了 RTCP 的协议格式和其主要承担的责任。接下来我们就进入实际工程部分的知识了。

我相信通过前面课程的学习，你对 H264 的码流结构已经较为熟悉了，H264 是在工程中用得比较多的编码标准，所以这里我们以 H264 为例来讲讲实际工程开发中，我们怎么将 H264 码流打包成 RTP 包。

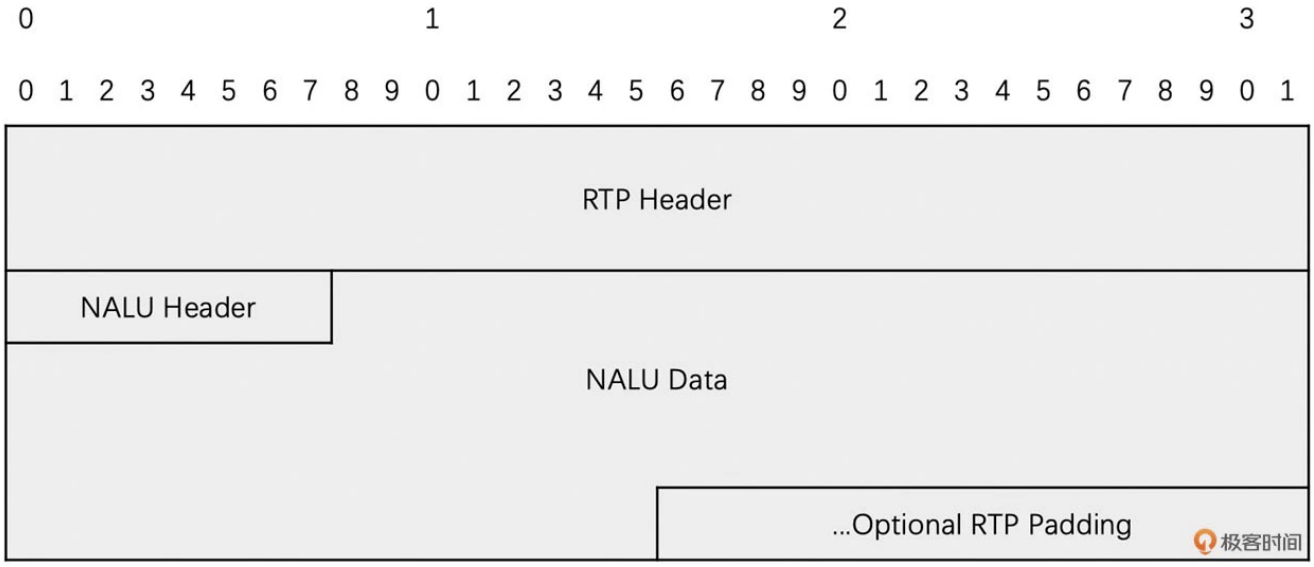
H264 RTP 打包

我们前面说了，H264 码流是放在 RTP 的有效载荷部分的。因此有效载荷前面的 RTP 头部跟码流本身是没有关系的，所以我们可以直接先将头部的字段填好就可以。接下来我们需要将 H264 码流填充到 RTP 有效载荷中去。

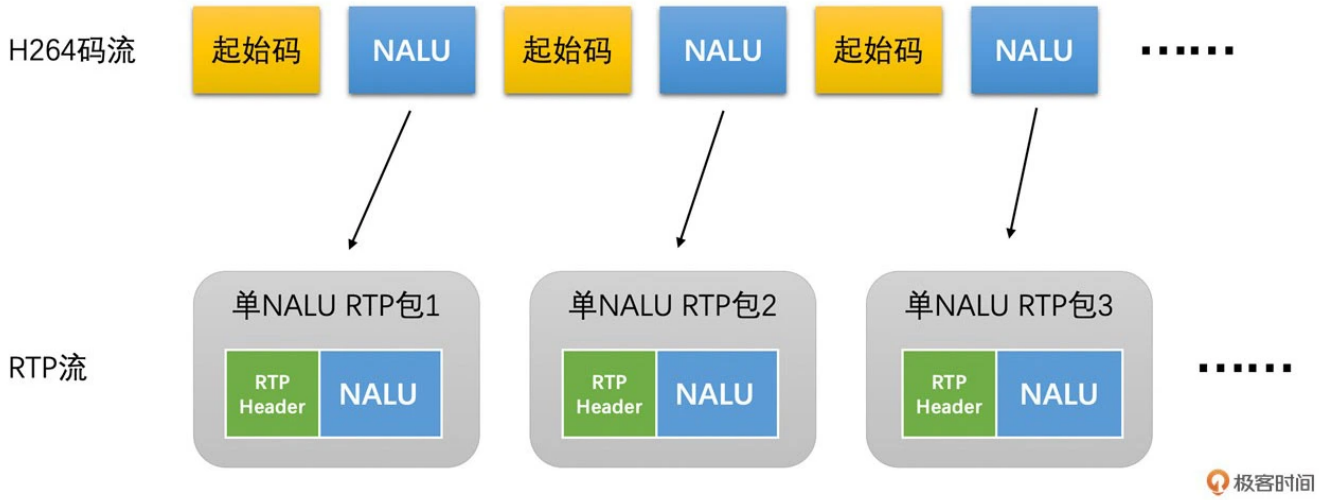
RTP H264 码流打包分为三种方式：分别是单 NALU 封包方式、组合封包方式、分片封包方式。顾名思义，单 NALU 封包方式是一个 NALU 打一个 RTP 包；而组合封包方式就是多个 NALU 打一个 RTP 包；分片封包方式则是一个 NALU 分开放在连续的多个 RTP 包中。下面我们来分别看一下各种打包方式是怎么样的。

1、单 NALU 封包方式

单 NALU 封包方式非常简单。我们在 RTP 头部的后面，直接放置 NALU 数据即可。注意，根据 RTP 的规定，这里需要**将 NALU 数据前面的起始码去除，**不要将起始码也带入 RTP 包中。其格式如下：



为了让你更直观地理解这种打包方式，我给出了打包的示意图。具体如下所示：



这种打包方式适合于单个 RTP 包小于 1500 字节（MTU 大小）的时候。一般来说，一些 P 帧和 B 帧编码之后比较小，就可以使用这种打包方式。

2、组合封包方式

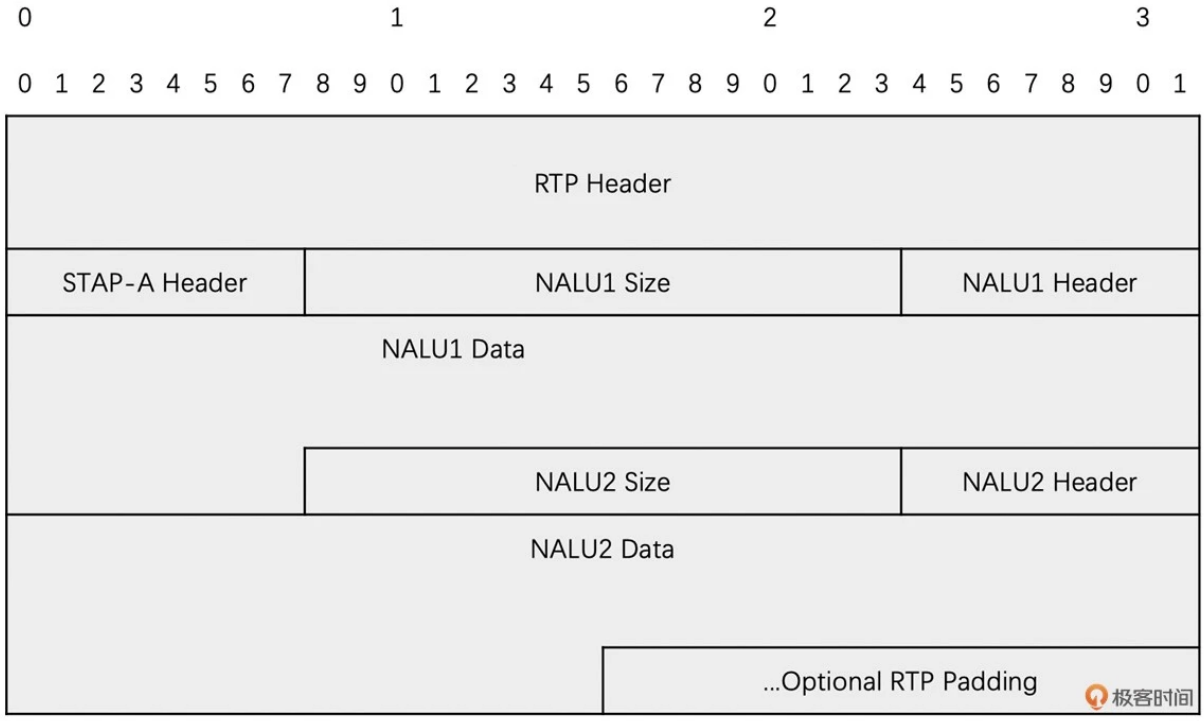
组合封包方式稍微复杂一些。它是将多个 NALU 放置在一个 RTP 包中。在 RTP 头部之后，且放置 NALU 数据之前，我们需要放置一个 1 字节的 STAP-A 的头部。其中，STAP-A Header 跟 NALU Header 的格式是一样的，只是 Type 字段的值不一样。因此，你可以参考 H264 码流结构课程中 NALU 小节来理解 STAP-A 的头部的格式。具体如下图所示：

其中，Type 的取值如下表所示。这里我需要提醒你一下，表中的 24 和 25 类型就是 STAP 组合封包方式。注意，我们这里只讲 STAP-A，这是因为 STAP-B 很少用到。

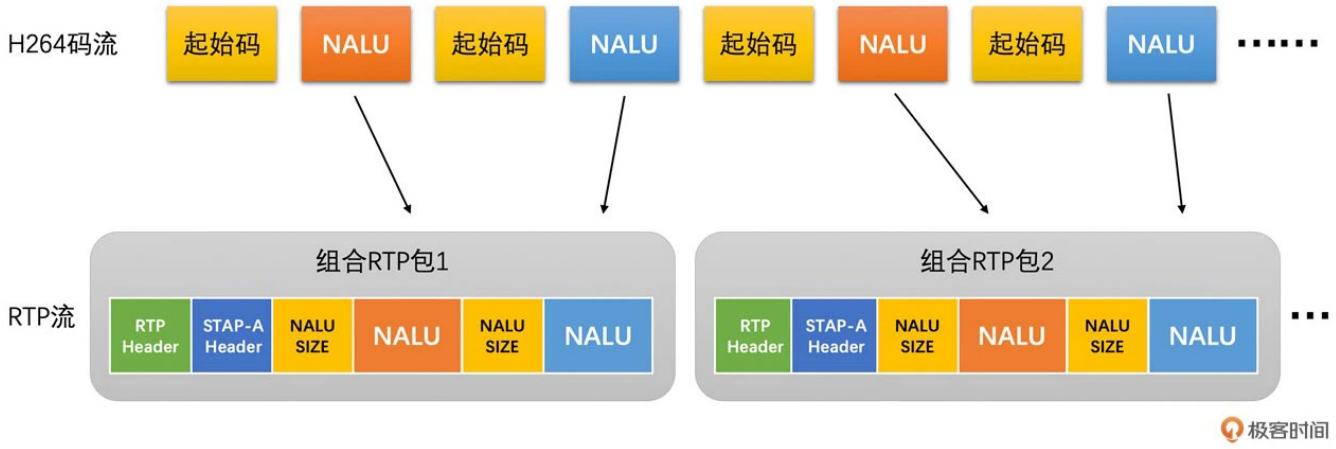
Type	类型	含义
24	STAP-A	单一时间的组合包
25	STAP-B	单一时间的组合包
26	MTAP16	多个时间的组合包
27	MTAP24	多个时间的组合包
28	FU-A	分片的单元
29	FU-B	分片的单元

极客时间

放置完 STAP-A Header 之后，在每一个 NALU 的前面我们需要放置一个 2 字节的 size 字段，用于表示后面的 NALU 的大小。之后才是 NALU 的数据。记住同样需要**去掉起始码**。其格式如下：



同样地，为了让你更直观地理解这种打包方式，我也给出了打包的示意图。具体如下所示：



这种打包方式适合于单个 NALU 很小的时候。因此，我们将多个 NALU 打包到一起也小于 1500 字节的时候就可以使用。但是由于一般多个视频帧加到一起还小于 1500 的情况比较少，所以视频数据的 RTP 打包一般来说用组合封包方式的情况也很少。

3、分片封包方式

分片封包就更复杂一些了，但却是我们经常用到的打包方式。

它是将一个 NALU 分开打包在连续的多个 RTP 包中。因此，我们首先需要一个 1 字节的 FU indicator 来表示当前 RTP 包是不是分片封包方式，再用一个 1 字节的 FU Header 来表示当前这个 RTP 包是不是 NALU 的第一个包，是不是 NALU 的最后一个包，以及 NALU 的类型。

为什么需要表示是不是第一个包以及是不是最后一个包呢？这是因为一个 NALU 被分开放在多个 RTP 包中，我们需要知道哪个是第一个 NALU 分片，哪个是最后一个 NALU 分片，以及哪些是中间分片。这样我们才能组成一个完整的 NALU。

那你可能会问，NALU 不是已经在 NALU Header 中有了 NALU Type 字段吗？为什么 FU Header 中还要有 NALU Type 呢？这是因为分片封包时需要去掉 NALU Header。因此，我们需要通过 FU Header 中的 NALU Type 得到 NALU 的类型。

其中，分片封装中的 FU indicator 跟 NALU Header 的格式也是一样的，也只是 Type 字段的值不同，所以我们可以参考组合封包小节中的表格。因为我们一般只使用 FU-A，所以接下来讲述的将是 FU-A 的分片封包方式。另外，FU Header 格式如下所示：

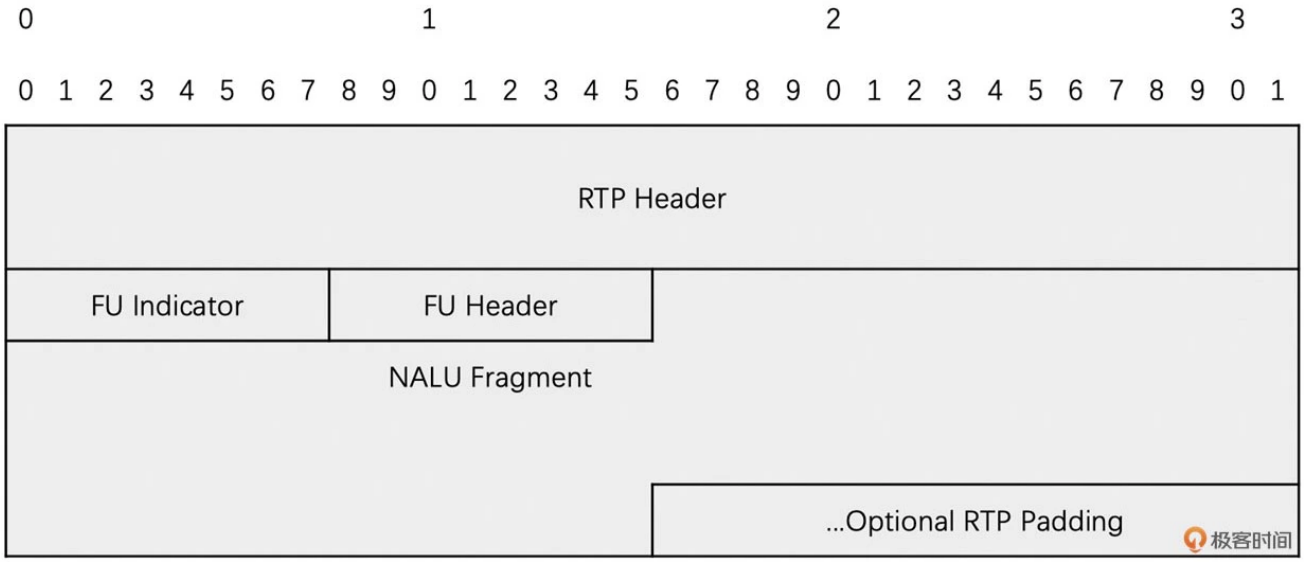
这里我简单解释一下各字段的含义：

S：起始位，占 1bit，为 1 则表示是 NALU 的第一个 RTP 包。

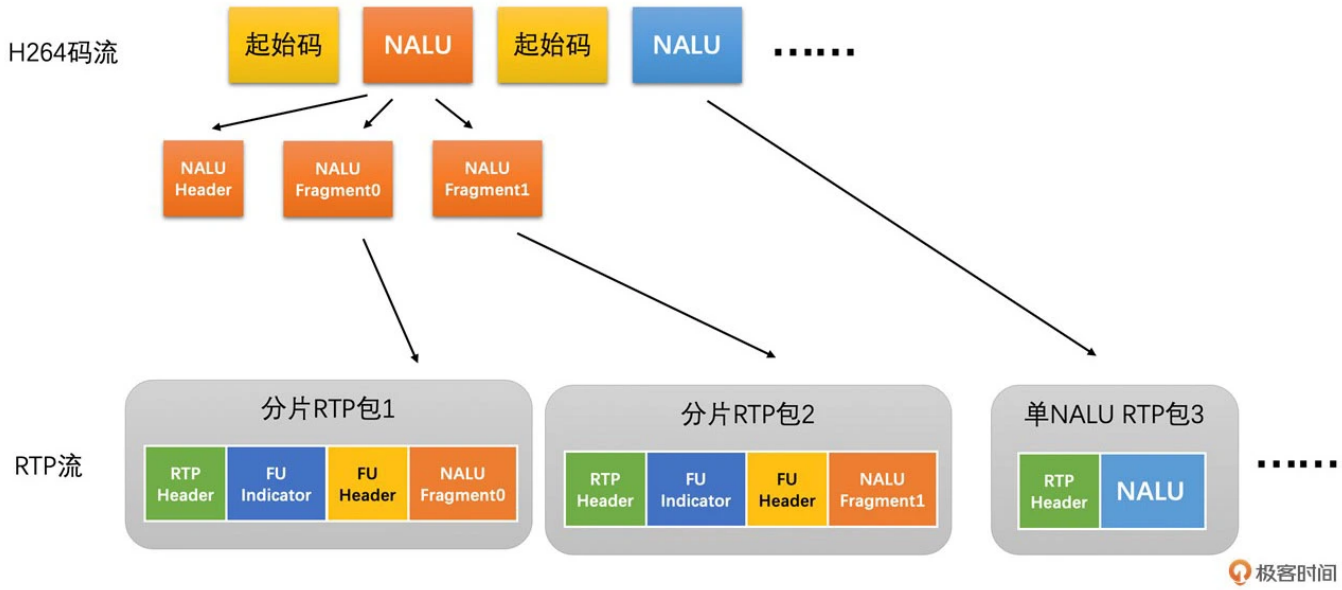
E：结束位，占 1bit，为 1 则表示是 NALU 的最后一个 RTP 包。

R：预留位，占 1bit。

Type：占 5bits，表示 NALU 类型。
分片打包的格式如下：



分片打包的示意图如下：



这种打包方式主要用于将 NALU 数据打包成一个 RTP 包时大小大于 1500 字节的时候，这是经常使用的视频 RTP 打包方法。

好了，以上就是三种打包方式。我们怎么选择使用哪种方式打包呢？一般来说，我们在一个 H264 码流中会混合使用多种 RTP 打包方式。一般来说，对于小的 P 帧、B 帧还有 SPS、PPS 我们可以使用单个 NALU 封包方式。而对于大的 I 帧、P 帧或 B 帧，我们使用分片封包方式。当然，你可以根据实际情况进行选择。

小结

好了，以上就是这节课的主要内容。接下来我们来总结一下。

首先，我们一起讨论了 RTP 协议和 RTCP 协议的主要作用。RTP 协议用来封装音视频数据，并且将音视频数据和一些基本信息打包到 RTP 包中传输到接收端。而 RTCP 协议则辅助 RTP 协议使用，其中一个主要的功能就是用来统计 RTP 包的发送情况，比如说丢包率和具体哪些 RTP 包在网络发送的过程中丢失了。RTCP 包将这些信息收集起来发送给 RTP 包的发送端。

然后，我们说明了 RTP 和 RTCP 协议是带宽预测和拥塞控制的基础，并且重点强调了 RTCP 协议本身只统计信息，而带宽预测和拥塞控制算法是需要我们自己实现的，RTCP 协议本身并没有这个功能。

最后，我们介绍了 H264 的 RTP 打包方式，总共有三种，分别是单 NALU 封包方式、组合封包方式和分片封包方式。

单 NALU 封包方式，一般适合 NALU 大小比较小，且打包出来的 RTP 大小小于 1500 字节的时候使用。

组合封包方式，适合多个 NALU 都很小，且合并在一起打包的 RTP 包小于 1500 字节的时候使用。

分片打包，则适合 NALU 比较大的情况，且打包成一个 RTP 包其大小会大于 1500 字节的时候使用。
这几种打包方式不是说只能选择一种，在一个 RTP 流中是可以存在多种打包方式的，即可以混合使用。

最后再一次强调，这节课和 H264 码流结构那节课都是非常重要的。它们在实际视频开发的过程中会经常用到，希望你可以熟练掌握。

思考题

为什么我们在选择 RTP 打包方式的时候，需要根据 NALU 大小是不是大于 1500 字节（MTU）来选择？

欢迎你在留言区和我分享你的思考和疑惑，你也可以把今天所学分享给身边的朋友，邀请他加入探讨，共同进步。下节课再见。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 08 | 变换量化：如何减少视觉冗余？

精选留言 (2)

写留言

狂奔的带刀蜗牛
2021-12-10

不超过1500主要是因为Udp协议的MTU为1500，超过了会导致Udp分片传输，而分片的缺点是丢了一个片，整包数据就废弃了

1

木希
2021-12-10

江哥，您这课程介绍的视频技术原理性都很详细，可以很好的理解视频原理性的东西，后面是否会带上部分代码逻辑的实现，原理跟实现还是有很长的路。

展开

共 1 条评论 >